Multiparty computation unconditionally secure against $\mathcal{Q}^2$ adversary structures

Adam Smith* Anton Stiglic†

December 15, 1998

Abstract

We present here a generalization of the work done by Rabin and Ben-Or in [16]. We give a protocol for multiparty computation which tolerates any $\mathcal{Q}^2$ active adversary structure based on the existence of a broadcast channel, secure communication between each pair of participants, and a monotone span program with multiplication tolerating the structure. The secrecy achieved is unconditional although we allow an exponentially small probability of error. This is possible due to a protocol for computing the product of two values already shared by means of a homomorphic commitment scheme which appeared originally in [8].

Keywords: Multiparty computation, general adversary structures, span programs, verifiable secret sharing.

1 Introduction

1.1 Multiparty computation

Multiparty computation (MPC) is a cryptographic task that allows a network of participants to emulate any trusted party protocol. Each player P_i starts with a private input x_i. The players run a protocol to compute some function $g(x_1, \ldots, x_n)$. The result of this function can then be revealed publicly or privately to some particular player. The protocol is deemed secure if cheating parties can obtain no more information from running the protocol than they would in the trusted party scenario (in which each player gives x_i to some external trusted party who then computes g and sends the result to all the relevant players). Goldreich, Micali and Wigderson proved that to accomplish MPC it is sufficient to always have the value of g revealed publicly and to assume that g is given by an arithmetic circuit (i.e. addition and multiplication gates) from K^n to K where K is some finite field.

The first general solution to this problem was given in [13]. They present a protocol for MPC which is secure under the assumptions that trapdoor one-way permutation exists, that the participants are restricted to probabilistic polynomial time (computationally bounded) and that the number of cheating parties is bounded above by t where $t < n/2$. In the situation where the participants can only cheat passively (i.e. by eavesdropping) they can remove the last assumption. In [2] and [7], the assumption of computational boundedness is removed and replaced by the assumption that each pair of players is connected by an authenticated secure channel. In this (non-computational) model they prove that MPC is possible with active adversaries if and only if $t < n/3$ and with passive adveraries if and only if $t < n/2$.

These results were extended in [16] to the scenario in which a reliable broadcast channel is also available. In that case active and passive cheaters can be tolerated if and only if $t < n/2$. However, to attain these bounds an exponentially small probability of error was introduced.

The result of [16] was first extended to more general adversary structures by Hirt and Maurer in [14]. However, maintaining an exponentially small probability of error entailed a superpolynomial loss of efficiency.

*School of Computer Science, McGill University, Montréal (Québec), Canada, asmith@cs.mcgill.ca

†Département d'Informatique et R.O., Université de Montréal, Montréal (Québec), Canada, stiglic@iro.umontreal.ca

We present a more efficient version of an extension of the [16] protocol using monotone span programs, following the ideas of [12] [1]. The relevant definitions as well as a precise statement of our results are presented in the remainder of this section.

1.2 Adversary structures and monotone functions

Given a set of players P, an adversary structure $\mathcal{A}$ over P is a set of subsets of players which is downward-closed under inclusion:

$$(B \in \mathcal{A} \text{ and } B' \subseteq B) \Longrightarrow B' \in \mathcal{A}.$$

Normally such a structure is used to represent the collection of all coalitions of players which a given protocol can tolerate without losing security: as long as the set of cheating players is in $\mathcal{A}$, the cheaters cannot breach the security of the protocol.

Classically, protocols such as those of [16] have tolerated *threshold structures*, which are of the form $\mathcal{A} = \{B \subseteq P : |B| \leq t\}$ for some t. However, [14] extends several of these results to more general structures, using the following definition:

Definition 1 *An adversary structure $\mathcal{A}$ over P is said to be $\mathcal{Q}^{\mathbf{k}}$ if no k sets in $\mathcal{A}$ add up to the whole set P, that is*

$$\nexists B_1, B_2, \ldots, B_k \in \mathcal{A} : \quad B_1 \cup B_2 \cup \cdots \cup B_k = P.$$

Hirt and Maurer ([14]) extended the results of [2, 16] (see section 1.1) which held for $t < n/3$ and $t < n/2$ to $\mathcal{Q}^3$ and $\mathcal{Q}^2$ structures respectively.

1.2.1 Monotone functions

Definition 2 *For a partial order $\leq$ on sets A and B, we say that a function $f : A \to B$ is* **monotone** *if for $x, y \in A$ we have*

$$x \leq y \Longrightarrow f(x) \leq f(y)$$

We can define a partial order on $\{0,1\}^n$ by the rule "$\mathbf{x} \leq \mathbf{y}$ iff each coordinate of $\mathbf{x}$ is smaller than the corresponding coordinate of $\mathbf{y}$." Then a function $f : \{0,1\}^n \to \{0,1\}$ is *monotone* if

$$\mathbf{x} \leq \mathbf{y} \Longrightarrow f(\mathbf{x}) \leq f(\mathbf{y}).$$

By identifying $\{0,1\}^n$ with $\wp(\{1,\ldots,n\})$, the relation $\leq$ on $\{0,1\}^n$ corresponds to inclusion ($\subseteq$) in $\wp(\{1,\ldots,n\})$. Then a monotone function f corresponds to a function from $\wp(\{1,\ldots,n\})$ to $\{0,1\}$ such that $A \subseteq B \Longrightarrow f(A) \leq f(B)$.

A monotone function f naturally defines an adversary structure $\mathcal{A}_f = \{B \subseteq P : f(B) = 0\}$.

Given an adversary structure $\mathcal{A}$ and a monotone function f, we say f *rejects* $\mathcal{A}$ if $f(B) = 0$ for all $B \in \mathcal{A}$, that is if $\mathcal{A} \subseteq \mathcal{A}_f$.

With these definitions in hand we can state the complexity of the Hirt-Maurer protocols: their generalization of [16] runs in time $m^{O(\log\log m)}$, where m is the size of the smallest monotone formula consisting of majority-accepting gates which rejects the adversary structure $\mathcal{A}$.

1.3 Monotone span programs

Span programs were introduced as a model of computation in [15]. They were first used as a tool for multiparty computation by Cramer, Damgård and Maurer [12]. In this section we define the concepts related to monotone span programs relevant to this paper.

Definition 3 *A* **monotone span program** *(MSP) over a set P is a triple (K, M, ψ) where K is a finite field, M is a $d \times e$ matrix over K and $\psi : \{1,\ldots,d\} \to P$ is a surjective function.*

The MSP associates to each subset $B \subseteq P$ a subset of the rows of M: the set of rows l such that $\psi(l) \in B$. This corresponds to a linear subspace of K^e (the span of those rows). The monotone function $f : \wp(P) \to \{0,1\}$ defined by a MSP is given by the rule $f(B) = 1$ if and only if the "target vector" $\epsilon = (1,0,0,\ldots,0)$ is in the subspace associated with B. If we denote by M_B the submatrix of M formed of the rows l such that $\psi(l) \in B$ then we get that

$$f(B) = 1 \iff \epsilon \in Im(M_B^T).$$

Given a MSP computing f, there is a secret sharing scheme which tolerates the corresponding adversary structure $\mathcal{A}_f$. This scheme is explained in section 2.1.

The definition above is sufficient for "secret sharing"-type protocols such as VSS and for multiparty computations in which multiplication in the field is not necessary. For general MPC, however, we need a stronger notion.

Definition 4 ((due to [12])) *A MSP (K, M, ψ) is said to be* **with multiplication** *if there exists a vector $\mathbf{r}$ (called a "recombination vector") such that*

$$\forall \mathbf{b}, \mathbf{b}' \in K^e : \quad \langle \mathbf{r}, M\mathbf{b} * M\mathbf{b}' \rangle = \langle \epsilon, \mathbf{b} * \mathbf{b}' \rangle$$

[1] Results similar to those in this article have been found independently in [11]

where $\epsilon = (1, 0, \ldots, 0)$, $\langle \cdot, \cdot \rangle$ *is the standard inner product on* K^e *and for* $\mathbf{v} = (v_1, ..., v_d), \mathbf{w} = (w_1, ..., w_d)$, *we have* $\mathbf{v} * \mathbf{w} = (v_1 w_1, ..., v_d w_d)$.

In [12] it is proved that for any $\mathcal{Q}^2$ adversary structure $\mathcal{A}$, one can construct a MSP with multiplication which rejects $\mathcal{A}$. The MSP can be constructed so it is linear in the size of the smallest majority-accepting formula rejecting $\mathcal{A}$.

Note that a counting argument shows that not all families of $\mathcal{Q}^2$ adversary structures over n players (for $n = 1, 2, \ldots$) can be rejected by a family of MSP's with size polynomial in n.

See the open questions in section 6 for further discussion.

1.4 Previous work

This work follows the initiative of Cramer, Damgård and Maurer in [12] for adapting existing threshold protocols to general adversary structures using monotone span programs. In that paper the results of [13] and [2, 7] were adapted to $\mathcal{Q}^2$ and $\mathcal{Q}^3$ structures respectively. We state their genralization of [2, 7]:

Theorem 1 *Let* $\mathcal{A}$ *be a* $\mathcal{Q}^3$ *adversary structure and* π *some multi-party protocol agreed upon* n *players. Let* (K, M, ψ) *be a MSP* with multiplication *rejecting* $\mathcal{A}$ *and suppose* π *can be implemented in* s *steps with operations over* K.

Then there is a protocol for π *tolerating* $\mathcal{A}$ *which is information-theoretically secure and which has complexity polynomial in* $\log|K|, s$ *and the size of* M.

1.5 This article

In this paper we adapt the results of [16] to $\mathcal{Q}^2$ structures with information-theoretic security. As mentioned above, this had already been done by Hirt and Maurer in [14] without using MSP's. However, their protocol ran in time $m^{O(\log \log m)}$ where m is the size of the smallest monotone formula consisting of majority-accepting gates which rejects the adversary structure $\mathcal{A}$.

Our protocol on the other hand is polynomial in the size of the smallest MSP with multiplication rejecting $\mathcal{A}$. Since MSP's with multiplication are at least as efficient as majority-accepting formulae (proved in [12]), our protocol is more efficient than the [14] construction.

The main results are:

Theorem 2 *Let* $\mathcal{A}$ *be a* $\mathcal{Q}^2$ *adversary structure on* n *players and* (K, M, ψ) *be a MSP rejecting* $\mathcal{A}$. *Suppose a reliable broadcast channel and secure communication between every pair of players is available.*

There is a VSS scheme for n *players, tolerating* $\mathcal{A}$, *which has error probability* $\leq 2^{-k}$ *and which has complexity polynomial in* $\log|K|, n, k$ *and the size of* M.

Theorem 3 *Let* $\mathcal{A}$ *be a* $\mathcal{Q}^2$ *adversary structure and* π *some multi-party protocol agreed upon* n *players. Let* (K, M, ψ) *be a MSP* with multiplication *rejecting* $\mathcal{A}$ *and suppose* π *can be implemented in* s *steps with operations over* K. *Suppose a reliable broadcast channel and secure communication between every pair of players is available.*

Then there is a protocol for π *tolerating* $\mathcal{A}$ *which has error probability* $\leq 2^{-k}$ *and which has complexity polynomial in* $\log|K|, s, n, k$ *and the size of* M.

Our protocol follows the construction of [16]. We first present the basic secret sharing scheme using MSP as well as an information checking protocol. We then give a protocol for *weak secret sharing* which we use to build a protocol for *verifiable secret sharing*. Using these tools we present the protocol for general MPC. For our protocol to be efficient, we change the product checking protocol of [16] [2]. The protocol we give is polynomial in $\log|K|$ whereas that of [16] is $\Omega(|K|^2)$. We conclude with some open questions.

2 Secret Sharing and Information Checking

2.1 Secret Sharing

Given a MSP (K, M, ψ), we can define a secret sharing scheme which tolerates the adversary structure $\mathcal{A}_f$ induced by the MSP (see section 1.3). Recall that M is a $d \times e$ matrix over the field K and $\psi : \{1, \ldots, d\} \to \{1, \ldots, n\}$ is an arbitrary function.

Say the dealer has a secret $a \in K$. He extends it to an e-rowed vector by adding random field elements $\rho_2, \ldots, \rho_e$ to make a vector $\mathbf{a}_* = (a, \rho_2, \ldots, \rho_e)$. Let $\alpha = M\mathbf{a}_*$ and let α_A denote the elements of α with indices in A where $A \subseteq \{1, \ldots, d\}$. Then the dealer gives α_l to player $P_{\psi(l)}$. In the end, each P_i receives $\alpha_{\psi^{-1}(i)}$.

From now on this protocol will be referred to as SHARE(D, a) where D is the dealer.

[2] The protocol given here was published in a different context (computational proofs of knowledge) in [8]. We independently "discovered" a slightly different version in 1998.

Lemma 4 *SHARE is a secret sharing scheme secure against $\mathcal{A}_f$. That is, no coalition in $\mathcal{A}_f$ can learn any information about the secret but any set of players* not *in $\mathcal{A}_f$ can reconstruct it.*

Proof: See [12]. □

2.2 Information Checking

The protocol in this section is based on [16]. All computations are done over $F = GF(3^k)$ where k is the security parameter. We require that $3^k > |K|^d$. This allows the encoding of any set of shares from the secret scharing scheme induced by the MSP (K, M, ψ).

We will use in the sequel a Guaranteed Information Checking (GIC) protocol:

Pre: D has already sent INT his secret $s \in F$.

Post: INT is guaranteed that an honest R (D may be dishonest) will always (i.e with very high probability) accept his value for s. Moreover, no information about s is leaked as long as D and INT are honest.

Protocol: GIC-Generate$(D \to INT \to R, s)$

1. D makes $2k$ vectors (y_i, b_i, c_i) such that $b_i \in_R F - \{0\}, y_i \in_R F$ and $c_i = s + b_i y_i$. He sends s and the y_i to INT and sends the check vectors (b_i, c_i) to R.
2. INT picks a random set $I \subseteq \{1, \ldots, 2k\}$ such that $|I| = k$ and broadcasts I.
3. R broadcasts the check vectors (b_i, c_i) with $i \in I$.
4. D checks whether or not this is indeed what he sent to R. If so, he broadcasts his approval. If not, he creates a new triple (y, b, c) such that $c = s + by$ and $b \neq 0$. He sends y to INT and *broadcasts* the single check vector (b, c).
5. Based on what he has seen, INT "guesses" whether or not R will now accept his value. If D approved in the previous step then INT decides "YES" if and only if R's pairs all agreed with the values INT possesses. If D disapproved and created a new check vector, INT outputs "YES" if and only if $c = s + by$ actually holds.
6. If INT thinks his value will be accepted by (an honest) R he broadcasts his approval. If not, INT asks D to broadcast s.

Protocol: GIC-Authenticate$(INT \to R, s)$.

1. INT sends s along with either $\{y_i : i \notin I\}$ or y (depending on what occurred at step 4) to R.
2. R accepts if any one of the y_i's agrees with her corresponding pair (b_i, c_i), or if y agrees with (b, c).

Notice that at the end of GIC-Generate, INT is guaranteed (with high probability) that an honest R will accept his value should he send it to R later on. Also notice that if D and INT are honest, no other party will gain any information about s (including R).

Lemma 5 *The GIC protocols have error probability less than 2^{-k}.*

Proof: See [16]. □

3 Weak Secret Sharing

This WSS scheme comes (essentially) from [16].

Before describing the protocol note that we will refer to the WSS of a value a, by D, as $[a]_D^W$. A similar VSSed value will be denoted $[a]_D^V$ and a verifiably shared secret belonging to no particular player will be written $[a]^V$.

From now on we will always assume that the MSP being used is $\mathcal{Q}^2$, that is we assume that the adversary structure $\mathcal{A}_f$ induced by the MSP is $\mathcal{Q}^2$.

The WSS scheme is in two parts: the commitment protocol (WSS) and the opening protocol (WSS-OPEN).

Pre: The dealer D has a secret $a \in K$.

Post: D has shared a such that either

- The shares the honest players hold are consistent with a single value which D can reveal, or
- The shares are inconsistent and D will always be caught and disqualified during the WSS-OPEN protocol.

Moreover, an honest D will never be disqualified.

Protocol: WSS(D, a)

1. SHARE(D, a)
2. For every $i \neq j$: GIC-Generate$(D \to P_i \to P_j, \alpha_{\psi^{-1}(i)})$.

Notice this protocol guarantees P_i that at some later time he can transmit his share to P_j and she will be convinced that D indeed gave him $\alpha_{\psi^{-1}(i)}$.

Based on this protocol we can define a *weakly shared value* to be a value a which a dealer D has shared (not necessarily correctly) such that GIC has been run for every pair (P_i, P_j) with $i \neq j$.

We now give the opening protocol.

Pre: a is weakly shared by D.

Post: There is a single value a which D can reveal. All the honest players will output the same value, which will be either a or *null*. They will output *null* only if D has acted dishonestly in sharing or revealing the secret.

Protocol: WSS-OPEN$(D, [a]_D^W)$

1. D broadcasts the vector $\mathbf{a}_*$ he created during the SHARE protocol.
2. Each P_i runs GIC-Authenticate with P_j (Thus P_j obtains $\alpha_{\psi^{-1}(i)}$ if P_i is honest and rejects the value if P_i tries to cheat. In the end, an honest P_i will have obtained α_θ where $f(\theta) = 1$, so he can reconstruct the secret if the shares of honest players are consistant.)
3. If for any i such that P_j accepted P_i's value there is an inconsistency (i.e. $\alpha_{\psi^{-1}(i)} \neq M_{\psi^{-1}(i)}\mathbf{a}_*$) then P_j accuses D.
4. If the set of accusers is not in the adversary structure (i.e. $f(\{accusers\}) = 1$) then D is disqualified. Otherwise his value (that is the first coordinate of $\mathbf{a}_*$) is accepted.

Notice that D's cooperation is essential to opening the commitment and that if need be, D can open his WSS only to a single player P_i by having all players send information only to P_i.

Lemma 6 *For any MSP whose adversary structure $\mathcal{A}_f$ is $\mathcal{Q}^2$, the pair of protocols (WSS, WSS-OPEN) is an $\mathcal{A}_f$-secure weak secret sharing scheme with error probability exponentially small in k.*

3.1 Linear operations on weakly shared values

It is clear that a WSSed value can be multiplied by any constant λ: each INT multiplies his share by λ and each R multiplies each of his pairs (b, c) by λ. Denote such a multiplication by $[\lambda a]_D^W \leftarrow \lambda * [a]_D^W$.

To add two WSSed values belonging to the same dealer, each player adds his shares of the two secrets to obtain his share of their sum. Then do GIC-Generate$(D \rightarrow P_i \rightarrow P_j, \gamma_{\psi^{-1}(i)})$ where γ is the vector of shares of the sum. The result is a WSS of the sum. Denote this by $[a+b]_D^W \leftarrow [a]_D^W + [b]_D^W$.

Remark: If D does not commit a properly but does commit b properly, he will be caught as a cheater if he opens $a+b$. This yields a simple zero-knowledge proof that a value is correctly committed by WSS. Have D pick b at random and commit to b and then use the preceding protocol to obtain $[a+b]_D^W$. Then flip a coin and have him either open b or $a+b$ depending on the outcome. If he was badly comitted to a he will be caught with probability $1/2$. Repeat the protocol k times to get exponentially small probability of error.

4 Verifiable Secret Sharing

Verifiable secret sharing is a primitive introduced in [9]. The scheme we give comes essentially from [16].

Pre: The dealer D has a secret $a \in K$.

Post: D is committed to a unique value which can be efficiently recovered without his help. Moreover, each player has committed to his share by means of a WSS. The shares of all players at the top (VSS) level are consistent. The shares of honest players at the lower (WSS) level are consistent.

Protocol: VSS(D, a)

1. SHARE(D, a).
2. For $l = 1, .., d$ do WSS$(P_{\psi(l)}, \alpha_l)$.
3. For $j = 1, .., kn$ do:
   (a) D chooses $c^{(j)} \in_R K$
   (b) SHARE$(D, c^{(j)})$ (yielding a random vector $\mathbf{c}_*^{(j)}$ and shares $\gamma_1^{(j)}, \ldots, \gamma_d^{(j)}$)
   (c) For $l = 1, .., d$ do:
   - WSS$(P_{\psi(l)}, \gamma_l^{(j)})$
   - $[\gamma_l^{(j)} + \alpha_l]_{P_{\psi(l)}}^W \leftarrow [\gamma_l^{(j)}]_{P_{\psi(l)}}^W + [\alpha_l]_{P_{\psi(l)}}^W$
4. For $j = 1, .., kn$ do:
   (a) $P_{j \bmod n}$ flips a coin and broadcasts the result.
   (b) **Heads:** Let
   - $b = c^{(j)}$
   - $\mathbf{b}_* = \mathbf{c}_*^{(j)}$

- $\beta = \gamma^{(j)}$

Tails: Let

- $b = a + c^{(j)}$
- $\mathbf{b}_* = \mathbf{a}_* + \mathbf{c}_*^{(j)}$
- $\beta = \alpha + \gamma^{(j)}$

(c) D broadcasts $\mathbf{b}_*$.

(d) Each P_i checks if $\beta_{\psi^{-1}(i)} = M_{\psi^{-1}(i)}\mathbf{b}_*$. If not, P_i accuses D. D must then broadcast all information given to P_i, that is $\alpha_{\psi^{-1}(i)}$ and $\gamma^{(j)}_{\psi^{-1}(i)}$ for all j. P_i is removed from the VSS protocol (if D does not broadcast the requested information, he is deemed corrupt).

(e) For $l = 1, .., d$ do WSS-OPEN$(P_{\psi(l)}, [\beta_l]^W)$ (as long as $P_{\psi(l)}$ remains in the protocol).
If $P_{\psi(l)}$ gets caught in WSS-OPEN or if the value he reveals is inconsistent with the $\mathbf{b}_*$ broadcasted by D then he is deemed corrupt and is removed from the protocol. All his shares of a and of all the $c^{(j)}$ are then broadcasted by D.

5. If the set of participants who are removed from the protocol (at any step) is qualified (i.e. not in the adversary structure) or if D broadcasts inconsistent information then D is deemed corrupt and the VSS is considered failed. Otherwise the VSS is deemed a success.

Note that as long as no errors occur in the subprotocols, the only way for D to succeed in passing off inconsistent shares for a is to correctly guess all the coin flips. Since at least one player is honest at least k of the coin flips are fair and so the failure probability is below 2^{-k}.

Based on this protocol we can define a *verifiably shared value* to be a value a such that every player is committed to his share of a via WSS. Moreover, the shares of all players at the top level must be consistent as must the shares of honest players at the bottom (WSS) level.

The opening protocol is VSS-OPEN:

Pre: a is a verifiably shared value.

Post: All honest players output a.

Protocol: VSS-OPEN$([a]^V)$

1. Each player opens his WSS to his share of the secret.
2. a is reconstructed from any qualified set of players who opened their WSS succesfully or whose shares had been broadcast in the VSS protocol.

Notice that no false shares can be contributed since any bad WSS's would have been detected (with high probability) in the VSS protocol. Moreover, all honest players *will* reveal their secret correctly and so a qualified set of shares *is* available for reconstruction.

Also notice that D's participation is not necessary and that the OPEN protocol works for any verifiably shared secret.

Lemma 7 *For any MSP whose adversary structure $\mathcal{A}_f$ is $\mathcal{Q}^2$, the pair of protocols (VSS, VSS-OPEN) forms an $\mathcal{A}_f$-secure VSS scheme with error probability exponentially small in k.*

This proves theorem 2.

4.1 Linear operations on verifiably shared values

It is possible to perform linear operations on a verifiably shared value by performing the corresponding operations on the shares (committed to via a WSS). In the case of VSS, it is *not* necssary that the secrets being added belong to the same dealer or indeed to anyone at all.

4.2 Converting WSS to VSS

A shared value $[a]^W_D$ can be converted to $[a]^V_D$ by throwing away the check vector information of the GIC protocols and considering the WSS as a simple secret sharing. The VSS protocol can then be started from step 2. The cooperation of the dealer *is* required for converting his WSS to a VSS.

5 Multi-Party Computation

The protocol for computing a function $g(x_1, x_2, \ldots, x_n)$ where x_i is the input of P_i follows the basic outline of [2, 7, 16]. Before the computation begins, the players decide on an arithmetic circuit over K which computes g. Each player commits to his input via a VSS $[x_i]^V_{P_i}$. The players then evaluate the circuit gate by gate to eventually end up with $[g(x_1, x_2, \ldots, x_n)]^V$. This commitment is then opened publicly.

We already discussed how to achieve a multi-party computation for addition and multiplication by a constant, so all that is missing now is a multi-party multiplication protocol.

5.1 Checking a product

We start by describing a protocol VSS-CP used by a dealer to prove that three secrets $[a]_D^V, [b]_D^V$ and $[c]_D^V$ satisfy $ab = c$. This protocol replaces the [16] protocol that used the multiplication table of the field K. The [16] protocol is insufficient since it runs in time $\Omega(|K|^2)$ whereas we require a protocol polynomial in $\log|K|$. We require this since it may be that the only polynomial sized MSP's for a given adversary structure happen to be over large fields (see open questions in section 6 for further discussion).

The protocol given here appeared in [8, 4], for commitments based on the discrete logarithm problem. As it appears here it works for any homomorphic commitment scheme (i.e. one which allows addition of secrets).

Pre: The dealer has $[a]_D^V$, $[b]_D^V$ and $[c]_D^V$ where $ab = c$.

Post: Every participant, knowing only shares of $[a]_D^V$, $[b]_D^V$ and $[c]_D^V$, will be convinced (with very small probability of error) that $ab = c$.

Protocol: VSS-CP$(D, [a]_D^V, [b]_D^V, [c]_D^V)$

1. Repeat for j = 1,...,kn,
   (a) D chooses $b' \in_R K$ and computes $c' = ab'$.
   (b) D commits himself to b' and c' by computing
   - $[b']_D^V \leftarrow \text{VSS}(D, b')$
   - $[c']_D^V \leftarrow \text{VSS}(D, c')$

   (c) The participant $P_{j \bmod n}$ flips a coin:
   i. If **Heads**:
   - The participants open $[b']_D^V$.
   - They collectively compute $[ab' - c']_D^V \leftarrow b' * [a]_D^V - [c']_D^V$.
   - They open this commitment and check it is 0.

   ii. If **Tails**:
   - The participants collectively compute and open $[b + b']_D^V$.
   - They collectively compute $[a(b + b') - (c + c')]_D^V \leftarrow (b + b') * [a]_D^V - [c']_D^V - [c]_D^V$.
   - They open this commitment and check it is 0.

Analysis: If in fact $c = ab$, the verification in steps *1(c)i* and *1(c)ii* will always be successful. On the other hand, if $c \neq ab$, there are two cases. First, if in fact D chose $c' = ab'$, in step 1(c)ii the participants would find that $c + c' \neq ab + ab'$. Second, if $c \neq ab$ but $c' \neq ab'$, the verification in step 1(c)i would fail. The protocol VSS-CP thus provides a proof to each honest player with probability of error less then 2^{-k}. Moreover this proof is zero-knowledge in that the information revealed is never enough to reveal any information about c, a or b.

5.2 The Multiplication protocol

This protocol is the multiplication protocol for active adversaries in [12]. It assumes that the MSP being used has multiplication (see section 1.3 for details).

Pre: We have $[u]^V, [v]^V$.

Post: We obtain $[uv]^V$, that is uv is a verifiably shared secret.

Protocol: MULT$([u]^V, [v]^V)$ We will denote by μ and ν the vectors that share u and v respectively.

1. For $l = 1, .., d$ do
   - $[\mu_l]_{\psi(l)}^V \leftarrow [\mu_l]_{\psi(l)}^W$
   - $[\nu_l]_{\psi(l)}^V \leftarrow [\nu_l]_{\psi(l)}^W$
     (see section 4.2 for how to do this).
2. For $l = 1, ..., d$, do:
   - Player $P_{\psi(l)}$ computes $\omega_l := \mu_l \nu_l$
   - $[\omega_l]_{\psi(l)}^V \leftarrow \text{VSS}(P_{\psi(l)}, \omega_l)$
   - Run VSS-CP$(P_{\psi(l)}, [\mu_l]_{\psi(i)}^V, [\nu_l]_{\psi(i)}^V, [\omega_l]_{\psi(l)}^V)$, so as to prove to everybody that his commited value ω_l is in fact $\mu_l \nu_l$.
3. For $l = 1, .., d$ do
   - Collectively compute

     $$[uv]^V = r_1 * [\omega_1]_{\psi(1)}^V + \cdots + r_d * [\omega_d]_{\psi(d)}^V,$$

     where $\mathbf{r} = (r_1, ..., r_d)$ is the recombination vector (this is a simple linear combination and can be done as in section 4.1).

If at any stage of the computation, a participant is detected as beeing a cheater, he is excluded from the protocol. The only problem that may arise is in step 1, since the participation of the dealer is necessary to convert a WSS to a VSS.

If this ever happens, we simply reset the protocol to the input distribution stage, the remaining players open the cheaters' VSSed inputs and then they restart the protocol. This will prolong the protocol by a factor of at most n.

Since uv is now a verifiably shared secret, it can be efficiently opened by the honest players using VSS-OPEN.

Notice the protocol given has no inherent probability of error. Its probability of error is at most the sum of the error probabilities of its subprotocols. As each of these has error probability exponentially small in k and is executed polynomially many times, the error probability remains exponentially small in k.

This completes the proof of theorem 3.

6 Open questions

1. It has not yet been proven whether MSP's *with multiplication* are super-polynomially more efficient than majority accepting circuits (or formulae) for computing max-$\mathcal{Q}^2$ functions[3]. In [12], a super-polynomial gap has been proved for general MSP's (which don't necessarily have the multiplication property). An interesting question is to determine how MSP's with multiplication perform as compared to MSP's without multiplication.

   One thing which is known is that schemes based on MSP's with multiplication (or even with strong multiplication) are at least as efficient as those based on threshold formulae. In particular this implies that the protocol given here is at least as efficient as the one in [14]. This is proved in [12].

2. No results have so far been published which extend threshold results other than [13], [2], [7] and [16]. Although it would seem that the ideas from [12] extend more or less directly to many distributed threshold protocols, this is not the case for asynchronous protocols.

   In asynchronous systems, MPC is possible tolerating any active $\mathcal{Q}^3$ adversary structure (by extending results of [5, 3]). However, no polynomial-time protocol currently exists. The bottleneck is a primitive from distributed computing known as Byzantine Agreement (BA). Although an efficient asynchronous BA protocol exists for threshold structures with $t < n/3$ (from [6]), the proof of correctness given there does not carry over to general structures.

3. Very few results exist which connect the size of the field used in an MSP to the size of the matrix M. Although one can pass from K to a subfield with blowup only quadratic in the degree of the extension [10], it is possible that certain functions have polynomial size MSP's which work only over unmanageably large prime fields.

[3] A max-$\mathcal{Q}^2$ adversary structure is one to which no more sets can be added without it losing the $\mathcal{Q}^2$ property. A max-$\mathcal{Q}^2$ function is one whose associated adversary structure $\mathcal{A}_f$ is max-$\mathcal{Q}^2$

Acknowledgements

We would like to thank Claude Crépeau for his support and many helpful comments, as well as for suggesting this area of study, as well as Ronald Cramer and Ivan Damgård for their comments on the manuscript.

References

[1] *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, Chicago, Illinois, 2–4 May 1988.

[2] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In ACM [1], pages 1–10.

[3] M. Ben-Or, B. Kelmer, and T. Rabin. Asynchronous secure computations with optimal resilience (extended abstract). In *Proceedings of the Thirteenth Annual ACM Symposium on Principles of Distributed Computing*, pages 183–192, Los Angeles, California, 14–17 Aug. 1994.

[4] J. Boyar, D. Chaum, I. Damgård, and T. Pedersen. Convertible undeniable signatures. In A. J. Menezes and S. A. Vanstone, editors, *Advances in Cryptology—CRYPTO '90*, volume 537 of *Lecture Notes in Computer Science*, pages 189–205. Springer-Verlag, 1991, 11–15 Aug. 1990.

[5] G. Bracha. Asynchronous Byzantine agreement protocols. *Information and Computation*, 75(2):130–143, Nov. 1987.

[6] R. Canetti and T. Rabin. Fast asynchronous Byzantine agreement with optimal resilience (extended abstract). In *Proceedings of the Twenty-Fifth Annual ACM Symposium on the Theory of Computing*, pages 42–51, San Diego, California, 16–18 May 1993.

[7] D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols (extended abstract). In ACM [1], pages 11–19.

[8] D. Chaum, J.-H. Evertse, and J. van de Graaf. An improved protocol for demonstrating possession of discrete logarithms and some generalizations. In D. Chaum and W. L. Price, editors, *Advances in Cryptology—EUROCRYPT 87*, volume 304 of *Lecture Notes in Computer Science*, pages 127–141. IACR, Springer-Verlag, 1988, 13–15 Apr. 1987.

[9] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults (extended abstract). In *26th Annual Symposium on Foundations of Computer Science*, pages 383–395, Portland, Oregon, 21–23 Oct. 1985. IEEE.

[10] R. Cramer. Personal communication, Aug. 1998.

[11] R. Cramer, I. Damgård, S. Dziembowski, M. Hirt, and T. Rabin. Efficient multiparty computations with dishonest minority. To appear in EUROCRYPT '99. Original version written Oct. 1998.

[12] R. Cramer, I. Damgård, and U. Maurer. General secure multi-party computation from any linear secret-sharing scheme. Most recent version available from Ronald Cramer at http://www.inf.ethz.ch/personal/cramer, 1998.

[13] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, pages 218–229, New York City, 25–27 May 1987.

[14] M. Hirt and U. Maurer. Complete characterization of adversaries tolerable in general multiparty computations. In *Proc. ACM PODC'97*, pages 25–34, 1997.

[15] M. Karchmer and A. Wigderson. On span programs. In *Proceedings of the Eighth Annual Structure in Complexity Theory Conference*, pages 102–111, San Diego, California, 18–21 May 1993. IEEE Computer Society Press.

[16] T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In *Proceedings of the Twenty First Annual ACM Symposium on Theory of Computing*, pages 73–85, Seattle, Washington, 15–17 May 1989.